Chapter 1 - Introduction

The Graphics Magician Picture Painter is a set of graphics editors and machine language routines that help you easily put professional, state-of-the-art graphics in your own programs. Included are routines that help you draw and recreate very compact, multicolored pictures (great for use in adventure games and educational software), and easily mix text with graphics on the screen. **The Graphics Magician Picture Painter** can be used just for drawing and saving pictures, or it can be used as a programmers' tool. The graphics routines included in this package are being used in dozen of commercially marketed software packages, produced by many of the most well-known publishers in the industry. And with **The Graphics Magician** available on several leading micro-computers, designers will find that most of the graphics work done on one machine can be easily transferred to several others, saving long hours of duplicated work, and in some cases making software portable in ways never before possible.

Also available from Penguin Software is **The Graphics Magician Animator;** a set of programs that will help you easily create arcade-fast animation that can be controlled from your own programs. It includes a shape editor for designing fast "pre-shifted" and player-missile shapes, a path editor for defining paths, an animation editor that lets you choreograph your animation, and all the routines necessary to give you full control over every shape from your own programs.

When learning to use **The Graphics Magician Picture Painter,** start simply. First of all sit down with your manual AND your computer. This is computer software, not a book. You'll learn it best by using it while reading about it. Since **The Graphics Magician** has many features, it is tempting to want to understand them all immediately. To start, just try some scribbling with lines and fills. Test the "redraw" command to see how your picture is recreated. Then experiment with the different options, one by one.

Input Devices

The Graphics Magician Picture Painter requires use of a joystick for input while drawing pictures. As other input devices are supported, we will note so with inserts.

Backup Copies

The Graphics Magician is provided on a copyable disk, and all the routines are accessible by your own programs. We strongly recommend that the first thing you do is make a backup copy or two and store the original in a safe place.

A registration number is written on your **Graphics Magician** disk and stamped on the inside cover of your manual. If you call with questions regarding use of this product please be prepared to provide this number. Also remember to send in your registration card so that we may notify you of any new versions or updates.

Licensing

Any of the routines enclosed may be freely used in your own programs. If the routines or facsimiles appear in another product for sale, there is no fee, but we do require that a license be obtained from Penguin Software stating that you have permission to use the copyrighted routines. Note that **The Graphics Magician** is available for several different microcomputers. In writing the different versions, the routines were made as compatible as possible, so much of the work you do on one computer can be easily transferred to another, if you desire. Some of the basic transfer routines are included in this package. Also, please consider Penguin Software as a possible publisher for your works.

Getting Started

"Boot" your **Graphics Magician** disk (put it in the disk drive, and turn on the drive and the computer). When the menu screen is displayed, choose the DOS option by pressing "**D**". When the DOS menu is displayed, you'll be ready to make your backup copies of the master **Graphics Magician** disk. Have a blank disk or two ready, and remove **The Graphics Magician** master from the drive. Insert a blank disk, and choose "format disk" by pressing the letter in front of the option. You will be asked "which drive to format?" Respond **1**, then press "**Y**" to verify (because anything on the disk in drive 1 will be destroyed).

After you have formatted one or two disks, choose the "copy disk" option. You will be asked for the location of your source and destination disks (from and to) If you have only one disk drive, type "**1,1**" (omit the quotes). With more than one drive, you can copy from drive 1 to drive 2 by typing "**1,2**". Now put the master into the disk drive, and follow the prompts for inserting your source disk (master) and destination disk (copy). After swapping several times, you will have a duplicate of **The Graphics Magician** master and can store the original safely away.

Now format one or two more disks to be used as data disks. **The Graphics Magician** disk is close to full, so the pictures you create should be stored on a separate data disk Create these by using the format disk option. Be sure to start with a blank disk, or one with nothing important on it, as it will be erased.

When done copying and formatting, put a copy of the master disk in the drive and press "**B**" to get back to **The Graphics Magician.**

You will be presented with a list of choices in parentheses. This is the "menu" screen When done using any of the modules listed in the choices, you will always be returned to this page. Note that throughout this manual, for ease of reading and understanding, single key choices will be listed with their meanings. Most of the choices in **The Graphics Magician** require only a single keypress, but expressing them in the form "(**P**)icture editor" instead of "P" helps one follow the meanings of each a little more easily

The Graphics Magician Picture Painter comes ready to use on an Atari with one disk drive and at least 48K of RAM. If you have one drive, you will have to switch disks when you read and save your picture files to your data disk. and again each time you are done with a module and wish to return to the menu screen. If you have more than one disk drive, you can set the **The Graphics Magician** so that it automatically uses drive 2 for the data disk. To set this, choose "(**M**)odify data disk" from the menu. At the bottom of the menu screen the program will list where it expects the master and data disks

Now you're ready to go. Remember, don't try too much too fast. Try simple pictures, then experiment with the options, one at a time

Chapter 2 - Drawing Pictures

The picture drawing system is designed to let you create screen pictures that take a minimal amount of storage space. It uses Atari graphics mode "E", which displays 160 dots across the screen, and 192 dots down. Each dot can be one of four base colors, and each of the base colors can be selected by you to be one of 128 available colors Furthermore, you can change the base colors so that they are different in each of up to 96 horizontal zones across the screen. (Each pair of horizontal lines on the screen can use a different set of base colors.) Thus, for example, you can have the top of your screen display the colors violet, orange, red, and yellow; the middle display blue, green, white, and maroon; and the bottom display black, brown, dark green, and turquoise. Of course any color combination is possible, with up to 96 of these horizontal zones. Besides the four base colors per zone, 71 colors have been made by putting together various dot patterns of the four base colors. Thus, instead of only four base colors in each zone, there are actually 71 blended colors.

Pictures always take 8K, approximately 8000 bytes, of display space in your computer However there are ways that allow you to take considerably less storage on disk. About 11 standard 8K pictures can fit on one side of a floppy disk. With **The Graphics Magician Picture Painter,** you can easily fit fifty to well over a hundred pictures on a single side of a disk.

Standard 8K pictures are stored as the values in the 8192 bytes that make up the graphics screen With **The Graphics Magician,** instead of storing the results of your drawing as a screen image, the moves that you make in creating your drawing are stored The moves for most drawings can be in hundreds of bytes instead of thousands. We call these "sequential pictures", since the sequence is remembered instead of the actual picture.

The effect this has is that the computer "remembers" what you do as you draw Later, when you want to view that picture again, the computer simply reconstructs your moves. very quickly. If you've played any graphic adventure games, you probably will recognize what this looks like; the picture redraws very rapidly before your eyes. What you see recreated are the moves that the artist made while drawing the picture the first time Most adventure games use this technique (many of them done with **The Graphics Magician**). since they demand that large numbers of pictures fit on a disk. There are also many educational products that use **The Graphics Magician** this way, since they also require a large number of graphic images.

There are four types of "moves" that you, the artist, can make. You may draw a line, fill an enclosed area with color, plot a computer "brush", or type a letter over your picture. In addition, you may choose from a palette of 71 color mixes, change the base colors in any zone of your picture, and select one of eight different brushes, ranging from a small, precision size to a large, airbrush effect.

Using the Picture Editor

From the menu, select "(**P**)icture editor".

Now you'll see a black screen, with a few text lines on the bottom, as in figure 2.1. If you move your joystick you'll see a small cursor in the shape of a crosshair move around the picture. This is what you control for drawing your picture.

```
'H' FOR HELP     BYTES USED = 00003
MODE  LINE  X=000  Y=000  FC=05  LC=3
                 000
```

Figure 2.1 - Picture Editor Command Lines

Line Mode

When you first start, you are in line mode. Pressing the **RETURN** key will put a "start line" command at the current location of your moving cursor. This is the starting point of the next line you draw. Pressing the joystick button draws a line from the start point to the movable cursor (ending point). The ending point also becomes the new starting point. Move your cursor around and try the effects of the **RETURN** key and joystick button.

How Long Is Your Picture?

When playing with line mode, note that three things happen on the bottom of the screen. The x,y position listed in middle of the second text line keeps changing as you move the cursor, and each time you press a button, the bottom line tells you what you just did ("Start Line at—", or "Draw Line to —"), and the byte count at the right side of the first text line tells you how many bytes you've used for your picture. Each "start line" or "draw line" command takes 3 bytes. Every picture has a starting length of 3 bytes

Deleting Steps

The first nice thing to learn is that pressing the **DELETE** key (in the upper right corner of the keyboard, you don't have to use the **SHIFT** key), will delete the last step. If you make a mistake, it's easy to back up as many steps as you want and try again. Note that each time you delete a step, you see how the redraw option works. The program remembers what you've done to that point and recreates everything except the step you deleted.

Fine Cursor Control

"(**Z**)ero" toggles the joystick input so the cursor moves more slowly, allowing you to zero in on a specific point more easily. Pressing "(**Z**)ero" again puts you back into normal mode.

Selection Page

Okay, now for some fun. Press the **SPACE BAR.** A graphic screen appears that shows your choices for modes across the top (line, fill, or brushes) and pictures of the eight brushes, two squares of four colors each on the left that show the current line and base colors, and a palette showing the 71 possible fill and brush colors. Two small boxes on the bottom say "range" and "normal". Your joystick controls a cursor that moves around the screen.

An "X" near the top should mark the line option, meaning that you are in line mode. Moving your cursor to the fill option or one of the eight brushes and pressing the button on the joystick will select that option and put you in a different mode.

A second "X" marks color 3 on the line color box on the left. This means that all your lines will be drawn in base color 3, which now is orange (the base color boxes are numbered as shown in figure 2.2). You can put your cursor on any of the four colors, and pressing the joystick button will select that for line color on new lines. Other line colors can be selected by changing any of the four base colors.

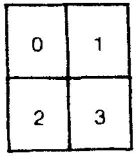


Figure 2.2 Base Color Numbering

The third "X" marks color 5 on the palette, dark blue. (The palette is numbered from left to right, starting with 0.) That is the current color used for filling and for brushes. Moving your cursor to any other color and pressing the joystick button selects that color for future fills and brushes.

Select fill mode and a fill color other than black, then press the **SPACE BAR.** The **SPACE BAR** switches between the drawing screen and the selection screen. While viewing the selection screen, you may change any of the options that you want, or you may just check the options and colors and press **SPACE BAR** again to get back to the drawing screen.

Fill Mode

When in fill mode, you can fill any enclosed area of base color 0 with the current fill color by positioning your cursor inside the area you want to fill and pressing the joystick button. Base color 0 is black when you start, but it can be changed. The area should be all base color 0, with borders in lines of any other color or the edge of the screen.

The fill routine is designed to be as fast as possible, so that pictures that are reconstructed in a finished program will appear very quickly. Most irregular areas will require two or three fill commands to fill the entire area, since with speed comes some compromise with completeness of fill. The fill routine used works the following way.

1) Scan directly up from the selected point until a border is found.
2) Move down one line, filling to the left and right borders.
3) Average the left and right borders to find the midpoint, and move down one line from there.
4) Check to see if the point moved down to is a border. If not, go back to step 2.

The basic thing to remember is step 1. It means that the best place to position your cursor is anywhere directly below the uppermost point in the area to fill. Using this one trick will minimize the number of fill commands necessary for filling any area.

Brushes

From the selection page, choose any of the eight brushes by pointing to it and pressing the joystick button. Go back to the drawing page and you'll see that your cursor is now in the shape of the brush that you selected. Each time you press the joystick button, the brush will be plotted with the color you selected.

The brushes give you a very large amount of control over detail, shading, and effects that cannot be achieved with "line and fill" coloring-book graphics. You will probably want to start most pictures by laying down a background with lines, then adding most of the colors with fills, and finally adding the detail touches with brushes.

There is no "brush up/brush down" selection that lets you cover a wide area. Each time you press the button the brush plots just once. If you want to move across an area, you have to keep pressing and releasing the button. While strange-seeming at first, remember that the computer is remembering your moves. If the brush were constantly down, it would have to remember each and every point that you move over, wasting a lot of memory very quickly.

Other Quick and Easy Options, Including Saving your Picture

While drawing, you have these other following choices available at all times. The letter commands are listed if you press "(**H**)elp".

The **ESC** key switches between graphics and text display (the normal mode, with the command lines at the bottom of the screen), and full-screen graphics mode. It has no effect on the picture itself, since the graphics area under the text is always available

Sometimes, with certain base color combinations, your command lines will not be easy to read. By pressing "**X**", you can turn off the interrupt routines that give all the extra colors, which will change all your picture colors, but make the command lines legible Pressing any other key will bring you back to the color mode.

"(**R**)edraw" will reconstruct the picture as it would be seen from a program. This is handy if you are trying for some animation (explained later), and most people admit that it s fun just to see what you have drawn redone at blinding speed by the computer

"(**S**)ave" allows you to save your picture in the special compact format created by **The Graphics Magician.** To view this picture later, you will have to load it back into the picture editor, or follow the instructions for using the PICDRAW routine in chapter 4

The "(**S**)ave" command only saves the displayed portion of the picture, which should be remembered when in edit mode, described below. This allows a way to extract parts of pictures, if desired.

"(**Control-S**)ave" (holding down the **CONTROL** key while pressing '**S**') saves a standard screen image of the picture displayed on the screen. This saves the full 8K screen area, and may be useful if you want to use the picture but not the PICDRAW routine

Note that there is NO way to edit a picture previously saved in 8K format with this picture editor. Pictures created with other graphics editors cannot be converted to the special compact format of **The Graphics Magician.** Since it is the moves that are saved, the pictures must be created with **The Graphics Magician Picture Painter** to have this compact format.

"(**Control-C**)lear" lets you clear the current picture and start over. Before clearing, the program will ask you to verify that you really want to clear the old picture. Press "(**Y**)es" if you do.

"(**Q**)uit" lets you return to **The Graphics Magician** menu. You are asked to verify that you really want to quit, in case you haven't saved the picture you are working on.

Changing Base Colors

Suppose you don't like our choices of black, green, blue, and orange for the base colors, or that you want to use different base colors for part of your picture. Go to the selection screen, and move your cursor to one of the boxes marked "bases". Press and hold down the joystick button, and move the joystick. The base color in the square that you chose will keep changing until you center the joystick or release the button. Note that the entire palette will change because of the different base color. You may change each of the base colors until you have the combination you want.

You're not done! Once you have the combination desired, move your cursor to the box marked "range" at the bottom, and press the joystick button. A third graphics page will be shown that gives the 192 horizontal lines on the screen and the four base colors chosen for each, which now should be the original black, green, blue, and orange. Suppose you want your new colors on the top half of the screen. Move your cursor to the top, press the button, then move to the middle of the screen (around 90), and press the button again. Presto! The top of the screen contains the new base colors. Press the **SPACE BAR** to go back to your palette screen, and once again to go to the drawing screen. The top half of any drawing you have will now be in the new base colors; the bottom will be in the original base colors.

Anytime you press **SPACE** to go to the palette screen, your cursor position will determine the base colors shown for the palette. The colors used at the cursor will be those shown on the palette. Note also that some combinations of base colors will make the command lines and the wording on the palette difficult to read.

It's suggested that you choose your base colors for the ranges of your picture before drawing the picture. You don't have to, and you can achieve some interesting effects by waiting until after all the drawing commands, but by choosing base colors early, you will always know the boundaries for various colors, and the full effects of each fill command

If, while changing base colors on the palette, you decide you want to start fresh with the original base colors we gave you, move your cursor to the "normal" box and press the joystick button. This does not affect the picture; it only resets the base colors on the palette.

Adding Text

You can add text to your picture at any time by positioning your cursor where you want the text to start and pressing "(**T**)ext". Whatever you type now will be overlaid onto your picture at that point. Note that color greatly affects clarity, and that the best results are with white text on a black background.

The text mode takes control of the cursor. If you want to leave text mode to delete a character or to make other changes, press **ESC.** Reaching the end of a line will also take you out of text mode.

You may reenter text mode at the last text cursor position by pressing "(**Control-T**)ext" while in normal drawing mode. Pressing "(**T**)ext" by itself always puts you in text mode at the position of the joystick cursor.

Loading Previously Saved Pictures

You may load in a picture that you had previously saved and add to it or edit it if you like. Press "(L)oad", and you will be asked if it's okay to clear the existing picture from memory. After pressing "(Y)es", type the name of the saved picture, and it will be loaded and redrawn.

Edit Mode

Since the picture editor saves pictures as a set of moves, it is possible to go back and edit those moves, much like a computer program itself. Edit mode allows you to single-step forward and backward through a set of picture commands, displaying what the picture looks like at each point, and allowing you to delete or add moves at any time. If you decide later that you don't like a color, find where you set that color and set a different one. If a house needs a few extra lines before the color is filled in, backstep to before the color was added and put in the lines. Easy!

Pressing "(**E**)dit" while in normal drawing mode will clear the screen to black and position you in your "picture program" at the first move. Each time you press the **right arrow** (you may press the key with the right arrow without using **CONTROL**), the next move in sequence will be displayed in words at the bottom of the screen and performed to the picture. Pressing the **back arrow** (again, **CONTROL** is not necessary) will back up one step. Your entire picture is still stored in memory, and pressing "(**R**)edraw" will bring it all back, but the edit mode allows you to add to or delete things that you did early in drawing your picture and see, step-by-step, how it was constructed.

In edit mode, all drawing commands remain usable, and anything you draw while in edit mode will be inserted into your picture. You can also use the **DELETE** key just as before to remove commands. Saving while in edit mode only saves what is visible on the screen. To be sure that you have the entire picture, just press "(**R**)edraw"

It is also possible to "back into" edit mode from the end of a picture by using the **back arrow.** Stepping forward through a picture is faster, but if you have a long picture and want to edit one of the last moves, this makes it easier.

"(**R**)edraw" will always let you out of edit mode by redrawing the entire picture stored in memory. You will also get out of edit mode if you single-step through the entire picture and reach the end.

"(**S**)ave", when used in edit mode, will only save the part of the picture that is displayed. This is a convenient way to save only the first half of a picture, for instance. If you want to save the whole picture, you should use "(**R**)edraw" to get out of edit mode

Chapter 3 - Tricks with Pictures

Objects

One of the features of many programs that require compact pictures is the ability to move objects from picture to picture, or to draw a picture with an object sometimes appearing, sometimes not. The obvious example is an adventure game, where something will appear in a picture, you take it with you, and thus it should no longer appear in the picture. We'll call this type of thing an "object".

Objects with **The Graphics Magician** are actually the same as pictures. You create them in generally the same way as you would any other picture. However, in your own program, when you use the picture-reconstructing "PICDRAW" routine, you tell it to draw that "picture" as an overlay. The picture thus becomes an object in the picture previously displayed.

Since objects are usually drawn over other pictures, you should be careful about what types of commands and colors you use. Remember that the object will always have the same base colors as the picture, or if you change the base colors in the object, the picture's base colors will also change.

With objects you should also be careful about when you use a fill command, if it is used at all. Since the fill command requires a base color 0 background, you must be sure this color is in place before you can fill. When you cannot be sure that the background picture will provide a base color 0 fill area, if you use a fill command you should first "black out" the area with one of the larger brushes.

For detail, most objects are done primarily with brushes and some careful use of lines Some prefer to set down a black (base color 0) background with brushes first, no matter what, so that all the drawing commands can be used freely. It's usually a matter of style

Creating your Object over a Background

To make it easier to choose colors and see how your object will look, you can draw it directly over a background picture. You may load a background by pressing "(**B**)ackground" while using the picture editor. If you load a background, you may draw your object directly on top of it. The background does not become part of your object, nor does it affect it in any way. Each time the screen is cleared, the background is displayed in place of the black screen that normally is shown.

If you loaded a background and then want to clear it, press "(**Control-B**)ackground" This will return you to the normal black background.

Animation with Pictures

One of the effects discovered after the original **Graphics Magician** was completed was that of animation created with the picture editor. Suppose you draw an entire picture, and in the picture is a man. Your commands in creating that picture are saved, so each time you view the picture, you'll see it redrawn. Now, suppose that once the picture is

complete, you draw more right on top of what you finished. For example, draw the man's eyes closed, then go back and draw them open again. When you press "(**R**)edraw", you'll see the man being drawn to completion, then his eyes will blink! You can extend it out further and keep drawing over and over the original picture and make all kinds of things in the room "animate".

This effect can be accomplished by drawing continuously on top of a picture, or it can be done by drawing a sequence of objects over your picture. The latter method has the advantage of allowing more control in timing, even tying it to user responses in your program. To do it with objects, you might load in the background of the man, then draw one object of his closed eyes, and another of his eyes open again. Or you might make one object his closed eyes quickly overlaid by his open eyes, since it's an immediate progression. In your program you'd draw the background picture, then whenever you wanted his eyes to blink, you would draw your blinking eye object(s). You could have it controlled by time, or have it happen every time someone touched a key. The flexibility is yours!

Chapter 4 - Using Pictures in your Programs

When you created your pictures in the picture editor, what was saved was your moves in drawing the picture. To display it, you need some way to tell the computer to recreate those moves. The machine language routine called PICDRAW does just that.

Put the PICDRAW routine on your disk

First, you must move the PICDRAW routine from **The Graphics Magician** disk to your program and data disk. To do so, go to DOS and use the option "O - duplicate file". To use PICDRAW with BASIC or any other cartridge, you must use the version "PICDRAWL" If you will be using PICDRAW with machine language and no cartridge, you may use "PICDRAWH", which gives you more room. (The "H" and "L" stand for "high" and "low", referring to their positioning in memory.)

Using PICDRAW

The following examples use PICDRAWL from BASIC. If you are using machine language, the calls will be summarized later, but the logic described here remains the same.

After PICDRAWL is moved to the same disk as your picture files, you can write a short BASIC program to display the pictures. We will start by giving you a few short subroutines that will allow you to access our machine language routines from BASIC These subroutines should be added to each program that used PICDRAWL. The following examples can be found on your disk under the file name PICTURE.BAS

Listing 1 is a quick routine that will initialize a few variables all the other routines will be using. Since we have to use machine language calls, the array CALL$ will contain the instructions that will execute each call. The instructions remain basically the same. For each routine, we'll just change the address of the subroutine by changing CALL$(3) and CALL$(4).

This initialize subroutine will also load an Atari DOS file by calling the Atari DOS loader routine. We'll use this to load PICDRAWL.

Picload.lst

```
5999 REM Initialize and Load PICDRAWL
6000 DIM FNAME$(15),CALL$(5)
6010 FNAME$="D:PICDRAWL.SYS" : FNAME$(15)=CHR$(155)
6020 CALL$(1)=CHR$(104) : CALL$(2)=CHR$(32) : CALL$(5)=CHR$(96)
6025 REM PLA, JSR ----', RTS
6030 ADDR=ADR(FNAME$) : ADDRH=INT(ADDR/256) : ADDRL=ADDR-ADDRH*256
     POKE 852,ADDRL : POKE 853, ADDRH
6040 CALL$(3)=CHR$(169) : CALL$(4)=CHR$(21)
6050 X=USR(ADR(CALL$)) : RETURN
```

Listing 1

In listing 1, line 6010 sets the name of the file we will load, line 6020 puts our machine language instructions into the variable CALL$, line 6030 pokes the address of the file name into DOS, line 6040 puts the DOS loader address into CALL$, and line 6050 calls the loader routine.

Listing 2 is a subroutine at line 6100 that will load a ".SPC" sequential picture file from the disk into the computer. Sequential picture files are loaded with a special loader in the PICDRAW routine that lets you load the file anywhere in memory. Before calling the subroutine in listing 2, you should put the file name to load in FNAME$. Line 6100 of the subroutine finds the address of the file name, line 6110 pokes that address into to PICDRAW loader routine, line 6120 tells PICDRAW where to load the file, and line 6130 calls the loader.

```
6099 REM Call Picture Loader for File FNAME$
6100 ADDR=ADDR(FNAME$) : ADDRH=INT(ADDR/256) : ADDRL=ADDR-ADDRH*256
6109 REM Save address of file name in PICDRAW address buffer
6110 POKE 29452,ADDRL : POKE 29453,ADDRH
6119 REM Set picture to load at $5000 hex, or 20480
6120 POKE 29454,0 : POKE 29455,80
6130 CALL$(3)=CHR$(PEEK(29442)) : CALL$(4)=CHR$(PEEK(29443)) : X=USR(ADR
     (CALL$)) : RETURN
```

Listing 2

Listing 3 is a subroutine that we will locate at line 6200 in BASIC. It calls the PICDRAW routine that sets the graphics display area to mode E and connects an "interrupt driver" that allows you to use different colors in horizontal zones across the screen.

```
6199 REM Set Graphics Display
6200 CALL$(3)=CHR$(PEEK(29440)) : CALL$(4)=CHR$(PEEK(29441)) : X=USR(ADR
     (CALL$)) : RETURN
```

Listing 3

Listing 4 is a subroutine that will clear the graphics screen and set the initial values for the PICDRAW routine.

```
6299 REM Clear Screen
6300 CALL$(3)=CHR$(PEEK(29446)) : CALL$(4)=CHR$(PEEK(29447)) : X=USR(ADR
     (CALL$)) : RETURN
```

Listing 4

Listing 5 is a subroutine we will put at line 6400 that draws the picture. This is the call to the actual picture drawing routine.

```
6399 REM PICDRAW
6400 CALL$(3)=CHR$(PEEK(29444)) : CALL$(4)=CHR$(PEEK(29445)) : X=USR(ADR
     (CALL$)) : RETURN
```

Listing 5

Listing 6 is the rest of our program. It initializes and loads the PICDRAW routine (line 10), lets you enter the name of a picture file (lines 20 and 30), then loads the file (line 40), sets the graphics mode (line 50), clears the screen (line 60), and draws the picture (line 70).

```
10 GOSUB 6000
20 DIM INAME$(8) : ? "FILE >"; : INPUT INAME$
30 FNAME$="D:" : FNAME$(3)=INAME$ : FNAME$(LEN(FNAME$)+1)=".SPC"
   FNAME$(LEN(FNAME$)+1)=CHR$(155)
39 REM Load picture into memory
40 GOSUB 6100
49 REM Set graphics display
50 GOSUB 6200
59 REM Clear graphics screen
60 GOSUB 6300
69 REM Call PICDRAW
70 GOSUB 6400
79 REM Wait for BREAK key
80 GOTO 80
```

Listing 6

Listings 1-6 make up the program PICTURE.BAS on your **Graphics Magician** disk. After you've drawn and saved a picture using the picture editor and moved the PICDRAWL routine to your data disk, you can turn off your computer, put in your BASIC cartridge, and boot on a standard DOS disk (not the **Graphics Magician** disk). When the screen says READY, put in your **Graphics Magician** disk and type LOAD "D:PICTURE.BAS" and press **Return.** When the disk stops, put your data disk in and type SAVE "D:PICTURE.BAS" and **Return**; now the program is on your own disk. Then type RUN and **Return.** The program will ask for your picture name, then load and display it. Press **BREAK** to get out of the program, (then reboot by turning the computer off and on).

Loading PICDRAWL into your Program - Important Note

Since the PICDRAWL loader that we use in listing 1 uses DOS, DOS must be on your disk or must have been loaded from another disk before you can load like this.

Using PICDRAW with Disk Access

Sequence is important, especially with the subroutine at 6200 that sets the graphics display and the interrupt drivers. You cannot use the disk drive while the interrupt drivers are connected. Once the graphics interrupt routines are in use, if you wish to access the disk again you must use:

```
POKE 54286,0
```

This will disconnect the interrupt routines, and it will have an effect on the colors displayed for your picture. After you are done with the disk access, you can reconnect the interrupt drivers with:

```
POKE 54286, 192
```

which will restore the interrupt routines and the graphics.

Redrawing a Picture without Reloading

Before being called, the PICDRAW routine must have the starting address of your sequential picture so that it can be reconstructed from the beginning. In our example, we poke in the starting address before we load the picture, in line 6120. This is necessary so that the routine knows where to load the picture, but is is also remembered later when calling the actual drawing routine. It is possible to have more than one sequential picture loaded into memory at any time, and to redraw whichever you want without loading from disk. To do so, though, you must use the same poke listed in line 6120 before redrawing.

The way the values to poke are computed is as follows: if the address of the start of the picture (where it was, or should be, loaded) is stored in variable ADDR,

```
POKE 29454, ADDR-INT(ADDR/256)*256
POKE 29455, INT(ADDR/256)
```

Putting an Object over a Picture

Suppose you created a picture and an object to be drawn over it. To do this from a program, you would follow the steps in the first example (listings 1-6), then disable the interrupt (POKE 54286,0), call the picture loader for your object picture (set FNAME$ to the name of the object, and call the subroutine at 6100), then skip the step that clears the screen. Instead, reenable the interrupt (POKE 54286,192), and call the redraw subroutine (at 6400). Instead of clearing the original picture, the object will just be drawn over it in the location in which the object was originally drawn.

Listing 7 can be used as a continuation of listings 1-6, and shows an example of drawing an object over a picture. This example uses an object named "WEREWOLF"

```
80 POKE 54286,0
90 FNAME$="D:WEREWOLF.SPC" : FNAME$(15)=155
100 GOSUB 6100
110 POKE 54286, 192
120 GOSUB 6400
130 GOTO 130
```

Listing 7

Chapter 5 - Advanced use of PICDRAW

Loading Groups of Pictures into One File

You can put several pictures in one long file, so that you only load one time, but can display several different pictures without going back to disk. To do so, you have to first understand the difference between the DOS loader and the PICDRAW loader. The DOS loader always loads a file to a specific, pre-determined location in memory. The PICDRAW loader will load a picture from disk to the address you specify just before loading.

To create a file with many pictures, you must first use the PICDRAW loader to load each file into memory, one right after another. The DOS loader can't do this, since each file would be loaded wherever it was originally saved from, and hence each would overwrite the last.

First, find the length of each picture you plan to use. This can be done by noting the "BYTES USED" in the picture editor for that picture. Suppose the results were as in figure 5.1.

Figure 5.1 - Sample Picture Lengths

Name	Length	Load At
House.SPC	1254	16384
Tree.SPC	879	17638
Moose.SPC	2318	18517

You next need to choose a starting location for your group of pictures. For example, we'll start at location 16384 ($4000). Now, load each of your pictures sequentially in memory. To do this you must use the PICDRAW loader in listing 2 of the previous chapter. For the first, you would load HOUSE.SPC at 16384.

For the second, add the length of "House" to 16384 to find the next available space 16384+1254=17638, so we load the second picture, in this example, at 17638 This would be TREE.SPC.

Remember the location at which you load each picture, since that's the address you must poke into locations 29454 and 29455 before redrawing the picture

The third file, MOOSE.SPC, is loaded at 17638+879=18517.

Now, compute the ending address by adding the length of the last picture (18517+2318=20835), and save the entire file by going to DOS and using option "K - BINARY SAVE". Give the new file name, the starting address (16384 in this example) and the ending address (20835 in the example). Do not give run or init addresses You now have a DOS format file with all the pictures you chose.

Using a Group of Sequential Pictures

To use this set of pictures in a program, you will use a program similar to that in listings 1-6. The difference is that you will not use the loader from PICDRAW (line 40 will not be a GOSUB 6100). Instead, repeat the DOS loader subroutine at line 6000, with your "pictures" file name substituted for "PICDRAWL.SYS" in line 6010. All else would be identical in the loader routine.

Then, before line 70, where you call the PICDRAW routine, you must poke the starting address of the picture you want into locations 29454 and 29455 (as described in the previous chapter). To draw a new picture, just loop back to 60 and call the clear routine, do the pokes for the new picture, and call PICDRAW again.

Chapter 5 - Advanced use of PICDRAW

Loading Groups of Pictures Into One File

You can put several pictures in one long file, so that you only load one time, but can display several different pictures without going back to disk. To do so, you have to first understand the difference between the DOS loader and the PICDRAW loader. The DOS loader always loads a file to a specific, pre-determined location in memory. The PICDRAW loader will load a picture from disk to the address you specify just before loading.

To create a file with many pictures, you must first use the PICDRAW loader to load each file into memory, one right after another. The DOS loader can't do this, since each file would be loaded wherever it was originally saved from, and hence each would overwrite the last.

First, find the length of each picture you plan to use. This can be done by noting the "BYTES USED" in the picture editor for that picture. Suppose the results were as in figure 5.1.

Figure 5.1 - Sample Picture Lengths

Name	Length	Load At
House.SPC	1254	16384
Tree.SPC	879	17638
Moose.SPC	2318	18517

You next need to choose a starting location for your group of pictures. For example, we'll start at location 16384 ($4000). Now, load each of your pictures sequentially in memory. To do this you must use the PICDRAW loader in listing 2 of the previous chapter. For the first, you would load HOUSE.SPC at 16384.

For the second, add the length of "House" to 16384 to find the next available space. 16384+1254=17638, so we load the second picture, in this example, at 17638. This would be TREE.SPC.

Remember the location at which you load each picture, since that's the address you must poke into locations 29454 and 29455 before redrawing the picture.

The third file, MOOSE.SPC, is loaded at 17638+879=18517.

Now, compute the ending address by adding the length of the last picture (18517+2318=20835), and save the entire file by going to DOS and using option "K - BINARY SAVE". Give the new file name, the starting address (16384 in this example) and the ending address (20835 in the example). Do not give run or init addresses. You now have a DOS format file with all the pictures you chose.

Using a Group of Sequential Pictures

To use this set of pictures in a program, you will use a program similar to that in listings 1-6. The difference is that you will not use the loader from PICDRAW (line 40 will not be a GOSUB 6100). Instead, repeat the DOS loader subroutine at line 6000, with your "pictures" file name substituted for "PICDRAWL.SYS" in line 6010. All else would be identical in the loader routine.

Then, before line 70, where you call the PICDRAW routine, you must poke the starting address of the picture you want into locations 29454 and 29455 (as described in the previous chapter). To draw a new picture, just loop back to 60 and call the clear routine do the pokes for the new picture, and call PICDRAW again.

Memory Usage and Different Versions of PICDRAW

PICDRAWL

PICDRAWL is the version of PICDRAW to use with a cartridge installed, such as BASIC. It requires at least a 40K system. The PICDRAW and display list/interrupt routines reside at $7300-$7FFF (29440-32767). The graphics memory is located from $8010 to $9E0F (32784-40463). The cartridge is at $A000, so $9E10-$9FFF is free.

Your picture buffer (where the picture commands get loaded) may be placed anywhere there is room, but the usual place is directly below the PICDRAWH routine (we used a bit of overkill in our example by putting the buffer at $5000, giving over 8K of room, over four times what will usually be needed). To find the highest location you may use for the buffer, find the length of your longest picture (from the picture editor) and subtract it from 29440 ($7300), the starting location of PICDRAWL. Sometimes it is useful to actually use two picture buffers. You may want one for "room", or background, pictures, and another separate buffer for objects.

The following addresses are special locations in PICDRAW. Most have already been described in use by the program example in listings 1-6. All addresses are in low/high format, as described in the pokes for starting location in the previous chapter.

Hex Address	Decimal	Use
$7300-7301	29440-29441	Contains address of routine that sets graphics mode E and sets the interrupt drivers
$7302-7303	29442-29443	Contains the address of the PICDRAW loader program
$7304-7305	29444-29445	Contains the address of the redraw routine
$7306-7307	29446-29447	Contains the address of routine that clears the graphics screen and resets all the PICDRAW default variables
$7308-7309	29448-29449	Contains the starting address of the color table used by the interrupt routine.
$730A-730B	29450-29451	This contains the address of the graphics memory
$730C-730D	29452-29453	This is where you store the address of the name of the file that the PICDRAW loader will load
$730E-730F	29454-29455	This is where you store the address at which the file should be loaded by the PICDRAW loader, and where the PICDRAW routine will draw from
$7010	29456	Contains the status of the last load or draw. 0 if it worked, nonzero if an error.

Figure 5.2 - PICDRAWL Memory Map

PICDRAWH

PICDRAWH is the version to use on a 48K system with no cartridge installed. It resides $2000 bytes higher, as are the graphics page and buffers. You should add $2000 to each PICDRAWL address to use PICDRAWH.

Changing Graphics Memory

The graphics memory is automatically put at $8010 by PICDRAWL. You may, if you have reason, put it at $6010 by changing the pointer at $730A. PICDRAWH puts the graphics memory at $A010, with optional graphics memory at $6010 and $8010.

Chapter 6 - Transferring and Interpreting

Pictures can be transferred to and from the Atari and other types of computers. With this version of **The Graphics Magician** is a routine that will accept and interpret picture files transferred from an Apple computer. It requires a special routine on the Apple that will send the files, and a cable that connects the Apple joystick port to an Atari joystick port.

Pressing "(**T**)ransfer" from the main menu will switch to the transfer program. When you are ready to receive, press "(**R**)eceive", then give the name for the transferred file It will be given a suffix of ".PTX", for picture transfer/exchange format. After you press **Return,** the Atari will wait for the file to be sent, then save it to disk.

Go back to the main menu and select "(**I**)nterpret". The interpret program will let you take a binary (.SPC) file and change it into a text (.PTX) file, or change a .PTX file to a SPC binary file. You want to do the latter, so press "(**T**)ext to binary". You will be asked for the name of the input file and for a name to give the output .SPC file. Then you'll be asked for X and Y multiplication factors. Coordinates are not the same on each computer. Apple uses a 280 by 192 dot graphics screen. On the Atari, we're using a 160 by 192 dot screen Apple coordinates are multiplied by 100 when they are sent out. To interpret them in the same proportions, the Atari must divide the X by 175 and the Y by 100. You should enter these values. The file will then be transferred to Atari .SPC format.

You can now go to the picture editor and see the results of your transfer. Chances are that you will want to make color changes, expecially those that make use of the extra colors available on the Atari by using color zones. You may do this by just single-stepping through the picture and adding and deleting commands as needed

Picture Listings

Pressing "(**L**)ist picture" from the interpret program lets you list the commands used to create your picture. Normally, this will be of little use, but it may at sometime be of interest to see your picture "program" listed.

Chapter 7 - Advanced Programming: The Interrupt Routines

If you want to add an interrupt routine of your own from machine language, and you want the color interrupt routine to keep functioning, you must put the following commands into your interrupt routine.

```
PHA                 ;push accumulator at start of interrupt
STX COLORX          ;save x register
LDX VCOUNT          ;get vertical line counter
LDA COLORB,X        ;these two statement only waste time
BNE * + 2           ;they can be replaced
LDA COLOR0-15,X     ;get playfield color 0
STA COLPF0
LDA COLORB-15,X     get background color
STA COLBAK
LDA COLOR1-15,X     ;get playfield color 1
STA COLPF1
LDA COLOR2-15,X     ;get playfield color 2
STA COLPF2
LDX COLORX
PLA
RTI
```

COLORX is any location set aside by you for saving the x-register contents.

VCOUNT is the vertical line counter, location $D40B
COLBAK is the background color register, location $D01A
COLPF0 is the playfield 0 color register, location $D016
COLPF1 is the playfield 1 color register, location $D017
COLPF2 is the playfield 2 color register, location $D018

The color table starts at the location stored in $7308-7309. COLORB takes the first 96 bytes, COLOR0 takes the next 96 bytes, COLOR1 takes the next 96 bytes, and COLOR2 takes the last 96 bytes.

The "-15" is used because VCOUNT has a value of 15 on the first visible line of the screen, corresponding to what we want to be 0.

Please also note that if you are making your own display list, the address of the graphics screen is $8010 or $A010, depending on the version of PICDRAW chosen. The display list used in **Graphics Magician** follows:

```
HIRAM    = $8000    ;or $A000
BLANK8   = $70      ; 8 blank screen lines
BLANK7   = $60      ;7 blank screen lines
BLANK1   = $80      ;1 blank screen line + interrupt
EINT     = $8E      ;mode E interrupt
E        = $0E      ;mode E
RELOAD   = $4E      ;reload memory scan points + mode E
JMPWT    = $41      ;wait for vertical blank and JMP

DSPL     .BYTE BLANK8,BLANK8,BLANK7,BLANK1
         .BYTE RELOAD
         .WORD HIRAM + $10
         .BYTE EINT,E
         (add the above line 49 more times here, for 50 total)
         .BYTE EINT,RELOAD
         .WORD HIRAM + $1000
         .BYTE EINT,E
         (add the above line 43 more times here, for 44 total)
         .BYTE E,JMPWT
         .WORD DSPL
```

Note that every other line is interrupted, which allows us to change the screen color on every other line. Similar design must be used if you create your own display list.

Interrupts must be stopped by storing a $40 in NMIEN ($D40E) before any I/O is done, since I/O is also done during interrupts. Resume the interrupts by storing a $C0 in NMIEN.